AMENDMENTS TO THE CLAIMS

Please amend the claims as follows.

1. (Currently Amended) A method of dynamically allocating a variable in a tracing framework, comprising:

   allocating dynamic memory in the tracing framework, having a plurality of data chunks;

   placing at least one of the plurality of data chunks onto a free list;

   encountering an enabled probe of an instrumented program;

   performing an action associated with the enabled probe, based on encountering the enabled probe;

   allocating the at least one of the plurality of data chunks on the free list to store the variable and removing the at least one of the plurality of data chunks from the free list, wherein the variable is associated with the action;

   deallocating the at least one of the plurality of data chunks and placing the at least one of the plurality of data chunks on a dirty list; and

   cleaning the at least one of the plurality of data chunks on the dirty list using a cleaning procedure to place the at least one of the plurality of data chunks on the free list,

   wherein the dynamic memory comprises first dynamic memory from a first processor and second dynamic memory from a second processor, and

   wherein the cleaning procedure comprises:

      moving one of the plurality of data chunks from the dirty list to a rinsing list if dirty list is not empty,

      issuing a first cross-call to the first processor and the second processor,

      moving one of the plurality of data chunks from the rinsing list to a clean list if the rinsing list is not empty upon receiving a response to the first cross-call,

      issuing a second cross-call to the first processor and the second processor, and

      setting a consumer dynamic memory state to clean in response to the second cross-call.

2. (Currently Amended) The method of claim 1, further comprising:
   associating the dynamic memory with [[a]] the consumer dynamic memory state, wherein the consumer dynamic memory state is associated with a tracing consumer, and wherein the tracing consumer is associated with the tracing framework.

3. (Previously Presented) The method of claim 2, further comprising:
   setting the consumer dynamic memory state after searching for the at least one of the plurality of data chunks to allocate.

4. (Original) The method of claim 2, wherein the consumer dynamic memory state is set to empty if all of the plurality of data chunks are allocated.

5. (Original) The method of claim 2, wherein the consumer dynamic memory state is set to dirty if all of the plurality of data chunks are either allocated or on the dirty list.

6. (Currently Amended) The method of claim 2, wherein the consumer dynamic memory state is set to rinsing if all of the plurality of data chunks are either allocated or on [[a]] the rinsing list.

7-8. (Cancelled)

9. (Currently Amended) The method of claim 1, further comprising:
   querying [[a]] the clean list for one of the plurality of data chunks if the free list is empty; and
   moving one of the plurality of data chunks from the clean list to the free list if the clean list is not empty.

10. (Original) The method of claim 1, further comprising:
   determining whether the variable has been previously allocated; and
   not allocating the variable if the variable has been previously allocated.

11. (Cancelled)

12. (Previously Presented) The method of claim 1, wherein the dynamic memory is associated with a tracing consumer, and wherein the tracing consumer is associated with the tracing framework.

13. (Previously Presented) The method of claim 1, wherein a size of the at least one of the plurality of data chunks is static.

14. (Original) The method of claim 1, wherein the dynamic memory is indexed using a hash table.

15. (Currently Amended) A system for dynamically allocating a variable, comprising:

a dynamic memory configured to store a plurality of data chunks;

a consumer dynamic memory state associated with the dynamic memory, configured to store a state of the dynamic memory; [[and]]

a tracing framework configured to allocate the variable to one of the plurality of data chunks using the consumer dynamic memory state, wherein the tracing framework comprises:

a free list configured to store at least one of the plurality of data chunks available for allocation, and

a dirty list configured to store at least one of the plurality of data chunks that has been deallocated; and

a cleaner configured to move one of the plurality of data chunks on the dirty list to the free list using a cleaning procedure,

wherein the variable is associated with an action,

wherein the action is associated with an enabled probe of an instrumented program, [[and]]

wherein the action is performed based on encountering the enabled probe,

wherein the dynamic memory comprises first dynamic memory from a first processor and second dynamic memory from a second processor, and

wherein the cleaning procedure comprises:

moving one of the plurality of data chunks from the dirty list to a rinsing list if dirty list is not empty,

issuing a first cross-call to the first processor and the second processor,

moving one of the plurality of data chunks from the rinsing list to a clean list if the rinsing list is not empty upon receiving a response to the first cross-call,
issuing a second cross-call to the first processor and the second processor, and
setting the consumer dynamic memory state to clean in response to the second cross-call.

16-18. (Cancelled)

19. (Currently Amended) The system of claim [[17]] 15, further comprising:
a tracing consumer configured to define a probe and a corresponding action

20. (Original) The system of claim 19, wherein the tracing framework is configured to dynamically allocate the variable in accordance with the action.

21. (Original) The system of claim 15, wherein the tracing framework is configured to set the consumer dynamic memory state after searching for at least one of the plurality of data chunks to allocate.

22. (Original) The system of claim 21, wherein the consumer dynamic memory state is set to empty if all of the plurality of data chunks are allocated.

23. (Original) The system of claim 21, wherein the consumer dynamic memory state is set to dirty if all of the plurality of data chunks are either allocated or on the dirty list.

24. (Currently Amended) The system of claim 21, wherein the consumer dynamic memory state is set to rinsing if all of the plurality of data chunks are either allocated or on [[a]] the rinsing list.

25. (Cancelled)

26. (Currently Amended) The system of claim 15, wherein the dynamic memory is indexed using a hash table.

27. (Currently Amended) A network system having a plurality of nodes, comprising:
a dynamic memory configured to store a plurality of data chunks;

a consumer dynamic memory state associated with the dynamic memory configured to store a state of the dynamic memory; [[and]]

a tracing framework configured to allocate a variable to one of the plurality of data chunks using the consumer dynamic memory state;

a free list configured to store at least one of the plurality of data chunks available for allocation;

a dirty list configured to store at least one of the plurality of data chunks that has been deallocated; and

a cleaner configured to move one of the plurality of data chunks on the dirty list to the free list using a cleaning procedure,

wherein the variable is associated with an action.

wherein the action is associated with an enabled probe of an instrumented program,

wherein the action is performed based on encountering the enabled probe,

wherein the dynamic memory executes on any node of the plurality of nodes,

wherein the consumer dynamic memory state executes on any node of the plurality of nodes, [[and]]

wherein the tracing framework executes on any node of the plurality of nodes,

wherein the dynamic memory comprises first dynamic memory from a first processor and second dynamic memory from a second processor,

wherein the free list executes on any node of the plurality of nodes,

wherein the dirty list executes on any node of the plurality of nodes,

wherein the cleaner executes on any node of the plurality of nodes, and

wherein the cleaning procedure comprises:

moving one of the plurality of data chunks from the dirty list to a rinsing list if dirty list is not empty,

issuing a first cross-call to the first processor and the second processor,

moving one of the plurality of data chunks from the rinsing list to a clean list if the rinsing list is not empty upon receiving a response to the first cross-call,

issuing a second cross-call to the first processor and the second processor, and setting the consumer dynamic memory state to clean in response to the second cross-call.

28-29. (Cancelled)